UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/749,735	12/30/2003	Vladimir Sachenko	6570P066	9043

8791 7590 03/17/2008

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
PHAM, LUU T	

ART UNIT	PAPER NUMBER
2137	

MAIL DATE	DELIVERY MODE
03/17/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/749,735	SACHENKO ET AL.	
	Examiner	Art Unit	
	LUU PHAM	2137	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 January 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-48* is/are pending in the application.

4a) Of the above claim(s) *21 and 35* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20, 22-34, and 36-48* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *28 January 2008* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to the Amendment filed on 01/28/2008.

2. Claims 1-48 are pending. In the instant Amendment, claims 1, 9, 13, 19-20, 22-34, 36-38, 44-45, and 48 are amended; claims 21 and 35 were cancelled; claims 1, 13, 20, 34, 41, and 45 are independent claims. This action is made **FINAL**.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. **Claims 20-33 are rejected under 35 U.S.C. 101** because the claimed subject matter does not belong to any of the four statutory categories set forth above.

- **Regarding claim 20**, although the preamble of the claim recites "*a computing apparatus,*" the body of the claim is directed to software implementation. Software does not fall into any of the four statutory categories set forth above; and therefore, the examiner maintains the claim rejection under 35 U.S.C. 101.

- **Regarding claims 21-33**, claims 21-33 are also rejected as nonstatutory under 35 U.S.C 101 as they do not recite any elements or hardware.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

4. **Claims 1-20, 22-29, 31-34, 36-38, and 40-48 are rejected under 35 U.S.C. 102(a)** as being unpatentable over Sun Microsystems (hereinafter "Sun"), "Building Web Services – Sun™ ONE Studio 5 programming Series," published in June 2003.

- **Regarding claim 1**, Sun discloses a method comprising:

accessing a description of a Web service *(page 29, WSDL; a link to WSDL file is given to make it available to create SOAP clients capable of issuing remove requests to web service; pages 112-122, sections Creating a Client From a WSDL File and Creating a Client From a UDDI Registry; WSDL file is located and parsed for creating a Web service client)*;

generating a Web service client proxy based, at least in part, on the accessed description of the Web service *(pages 112-122; sections Creating a Client From WSDL and Creating a Client From a UDDI Registry Entry; the default URL comes from the WSDL that is used to generate the client proxy)*;

providing a client protocol implementation for the generated Web service client proxy, wherein the provided client protocol implementation is to process a message exchanged between the Web service client proxy and the Web service *(pages 29 and 112-122; WDSL file describes external interface for a Web service; with this information, developers can create SOAP client capable of issuing remote requests to the Web service; tModels is published by web service provider to provide external interface to create web service client)*; and

setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy *(page 95-105; Figs. 3-7 and 3-8; user is able to set the client type, either JAX-RPC or kSOAP; user is able to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components)*.

- **Regarding claim 2**, Sun discloses the method of claim 1, wherein setting the feature of the client protocol implementation comprises:

selecting an authentication type for a client authentication protocol implementation to define an authentication type for the message exchanged between the Web service client proxy and the Web service *(pages 189-214; Fig. A-2, Authentication setup dialog box; JAX-RPC and IDE support HTTP Basic Authentication and HTTPS/SSL Authentication and Encryption; select the radio button for HTTP Basic Authentication)*.

- **Regarding claim 3**, Sun discloses the method of claim 2, wherein selecting the authentication type for the client authentication protocol implementation comprises selecting an X.509 certificate authentication type for the client authentication protocol implementation *(page 192; server and client authenticate each other by exchanging public key certificate, X.509)*.

- **Regarding claim 4**, Sun discloses the method of claim 1, wherein setting the feature of the client protocol implementation comprises:

selecting a HyperText Transport Protocol (HTTP) proxy for a client HTTP proxy protocol implementation to define an HTTP proxy for the message exchanged between the

Web service client proxy and the Web service *(pages 76-77; Fig. 2-29; user is able to select http (the default) or https for the Network Access Point Type using UDDI publish Web service dialog box).*

- **Regarding claim 5**, Sun discloses the method of claim, wherein setting the feature of the client protocol implementation comprises:

selecting a wrapper for a client wrapper protocol implementation to define a wrapper for the message exchanged between the Web service client proxy and the Web service *(page 36; Fig. 1-2; SOAP request is an XML wrapper that contains a method call and SOAP response is an XML wrapper that contains the return value).*

- **Regarding claim 6**, Sun discloses the method of claim 5, wherein selecting the wrapper for the client wrapper protocol implementation comprises:

selecting a header for a client header protocol implementation to define a header for the message exchanged between the Web service client proxy and the Web service *(pages 137-138; Code Example 4-5; sample Handler code to add and initialize a header to a SOAP message).*

- **Regarding claim 7**, Sun discloses the method of claim 6, wherein selecting the header for the client header protocol implementation comprises:

selecting a Simple Object Access Protocol (SOAP) header for a client SOAP header protocol implementation to define a SOAP header for the message exchanged between the Web service client proxy and the Web service *(pages 132-138; Code Example 4-1; SOAP header is defined and implemented).*

- **Regarding claim 8**, Sun discloses the method of claim 1, wherein setting the feature of the client protocol implementation comprises:

setting a session feature for a client session protocol implementation to define a session feature for the message exchanged between the Web service client proxy and the Web service (*pages 61-63; Figs. 2-15, 2-16, 2-17, and 2-18; session bean stateful can be set by using either Session EJB setup dialog box, Figs. 2-15 and 2-16, or modifying the session bean class to add methods named startOrder and submitOrder, Figs. 2-17 and 2-18).*

- **Regarding claim 9**, Sun discloses the method of claim 8, wherein setting the session feature comprises:

restarting a session between the Web service client proxy and the Web service *(pages 91-129; Creating a Web Service Client; Web service client connects with Web service server to obtain WSDL file and another EJB session will be established when executing the JAX-RPC Client created).*

- **Regarding claim 10**, Sun discloses the of claim 1, wherein accessing the description of a Web service comprises:

accessing a Web Service Description Language document describing the Web service *(page 29; WSDL, Web Service Description Language, is a W3C standard, XML-based language used to described a web service's external interface).*

- **Regarding claim 11**, Sun discloses the method of claim 1, wherein generating the Web service client proxy comprises:

generating a deployable Web service client proxy *(pages 95-99; Generating a JAX-RPC Client; JAX-RPC Client is known as a deployable Web service client)*.

- **Regarding claim 12**, Sun discloses the method of claim 1, wherein generating the Web service client proxy comprises:

generating a standalone Web service client proxy *(pages 110-111; Generating a kSOAP Client; kSOAP Client is known as standalone Web service client)*.

- **Regarding claim 13**, Sun discloses an application server *(Figs. 1-1, 1-2, and 1-3; Java client and Web container)* comprising:

a network interface to access a description of a Web service *(pages 31, 68, and 112; web service is deployed and available over network)*; and

a processor and logic executable thereon to generate a Web service client proxy based, at least in part, on the accessed description of the Web service *(pages 112-122; sections Creating a Client From WSDL and Creating a Client From a UDDI Registry Entry; the default URL comes from the WSDL that is used to generate the client proxy)*;

provide a client protocol implementation for the generated Web service client proxy, wherein the provided client protocol implementation is to process a message exchanged between the Web service proxy and the Web service *(pages 29 and 112-122; WDSL file describes external interface for a Web service; with this information, developers can create SOAP client capable of issuing remote requests to the Web service; tModels is published by web service provider to provide external interface to create web service client)*; and

configure a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy *(page 95-105; Figs. 3-7 and 3-8; user is able to set the client type, either JAX-RPC or kSOAP; user is able to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components)*.

- **Regarding claim 14**, Sun discloses the application server of claim 13, wherein the processor and logic executable thereon to configure a feature of the client protocol implementation comprises:

a processor and logic executable thereon to configure an authentication type for a client authentication protocol implementation to define an authentication type for the message exchanged between the Web service client proxy and the Web service *(pages 189-214; Fig. A-2, Authentication setup dialog box; JAX-RPC and IDE support HTTP Basic Authentication and HTTPS/SSL Authentication and Encryption; check the radio button for selecting HTTP Basic Authentication)*.

- **Regarding claim 15**, Sun discloses the application server of claim 13, wherein the processor and logic executable thereon to configure a feature of the client protocol implementation comprises:

a processor and logic executable thereon to configure a HyperText Transport Protocol (HTTP) proxy for a client HTTP proxy protocol implementation to define an HTTP proxy for the message exchanged between the Web service client proxy and the

Web service *(page 76; either http (the default) or https is configured for the Network Access Point Type)*.

- **Regarding claim 16**, Sun discloses the application server of claim 13, wherein the processor and logic executable thereon to configure a feature of the client protocol implementation comprises:

a processor and logic executable thereon to configure a wrapper for a client wrapper protocol implementation to define a wrapper for the message exchanged between the Web service client proxy and the Web service *(page 36; Fig. 1-2; SOAP request is an XML wrapper that contains a method call and SOAP response is an XML wrapper that contains the return value)*.

- **Regarding claim 17**, Sun discloses the application server of claim 13, wherein the processor and logic executable thereon to configure a feature of the client protocol implementation comprises:

a processor and logic executable thereon to configure a session feature for a client session protocol implementation to define a session feature for the message exchanged between the Web service client proxy and the Web service *(pages 61-63; Figs. 2-15, 2-16, 2-17, and 2-18; session bean stateful can be set by using either Session EJB setup dialog box, Figs. 2-15 and 2-16, or modifying the session bean class to add methods named startOrder and submitOrder, Figs. 2-17 and 2-18)*.

- **Regarding claim 18**, Sun discloses the application server of claim 13, wherein the application server is a Web application server *(Figs. 1-1, 1-2, and 1-3; Web services known as Web application server)*.

- **Regarding claim 19**, Sun discloses the application server of claim 18, wherein the application server is a JAVA 2 Enterprise Edition (J2EE) compatible application server *(page 55; assembling the J2EE application)*.

- **Regarding claim 20**, Sun discloses a computing apparatus comprising:

a client application to invoke a method of a Web service *(pages 91-122; Figs. 3-1 to 3-9; web client, created from either a local IDE web service, from WSDL, or from a UDDI registry entry, sends SOAP request to the web server)*;

a Web service client proxy coupled with the client application to expose the method of the Web service to the client application and exchange a message with the Web service *(pages 91-122; Figs. 3-1 to 3-9; JAX-RPC Client is created and deployed; SOAP messages are exchanged between Web server and Web client)*; and

a protocol implementation coupled with the Web service client proxy *(page 21; Sun One Studio 5 IDE to build web services, to make web services available to others through a UDDI registry, and to generate web service client from a local web service or a UDDI registry; page 58; the IDE provides comprehensive support for creating web service clients from any of the following sources: a local IDE web service, a WSDL file, a web service published in a UDDI registry)* to process a message exchanged between the Web service client proxy and the Web service *(pages 34-38 and 91-122; Figs. 1-1, 1-2, and 1-3; Web client sends SOAP request message to the Web server and receives SOAP response from the Web server)*, the protocol implementation comprising a security protocol implementation to provide a security service for the message *(pages 189-214; Fig. A-2;*

either HTTP Basic Authentication or HTTPS/SSL Authentication or Encryption is utilized to secure message exchanged between client and server).

Regarding claim 22, Sun discloses the computing apparatus of claim 20, wherein the security protocol implementation comprises an authentication protocol implementation to authenticate the message exchanged between the Web service client proxy and the Web service *(pages 189-214; Fig. A-2; either HTTP Basic Authentication or HTTPS/SSL Authentication and Encryption is utilized to secure message exchanged between client and server; user name and password are used for HTTP Basic Authentication; Public key encryption and Public key certificates are used for HTTPS/SSL Authentication and Encryption).*

- **Regarding claim 23**, Sun discloses the computing apparatus of claim 22, wherein the authentication protocol implementation is to implement a digital certificate protocol for the message *(pages 189-214; Public key certificates, digital certificates, are used for HTTPS/SSL Authentication and Encryption).*

- **Regarding claim 24**, Sun discloses the computing apparatus of claim 20, wherein the security protocol implementation is to implement an encryption protocol implementation to provide an encryption service for the message *(pages 189-214; Public key encryption and Public key certificates are used for HTTPS/SSL Authentication and Encryption).*

- **Regarding claim 25**, Sun discloses the computing apparatus of claim 20, wherein the protocol implementation further comprises a wrapper protocol implementation to provide a wrapper for the message *(page 36; Fig. 1-2; SOAP request is an XML wrapper*

that contains a method call and SOAP response is an XML wrapper that contains the return value.

- **Regarding claim 26**, Sun discloses the computing apparatus of claim 25, wherein the wrapper protocol implementation comprises a header protocol implementation to process a header for the message *(pages 137-138; Code Example 4-5; sample Handler code to add and initialize a header to a SOAP message).*

- **Regarding claim 27**, Sun discloses the computing apparatus of claim 26, wherein the header protocol implementation comprises a Simple Object Access Protocol (SOAP) header implementation to process a SOAP header for the message *(pages 137-138; Code Example 4-1, SOAP Header; Code Example 4-5, sample Handler code to add and initialize a header to a SOAP message).*

- **Regarding claim 28**, Sun discloses the computing apparatus of claim 20, wherein the protocol implementation further comprises a session protocol implementation to process a session between the Web service client proxy and the Web service *(pages 61-63; Figs. 2-15, 2-16, 2-17, and 2-18; EJB session bean is implemented for the Web service application; session bean stateful can be set by using either Session EJB setup dialog box, Figs. 2-15 and 2-16, or modifying the session bean class to add methods named startOrder and submitOrder, Figs. 2-17 and 2-18).*

- **Regarding claim 29**, Sun discloses the computing apparatus of claim 20, wherein the protocol implementation further comprises a HyperText Transport Protocol (HTTP)

proxy protocol implementation to establish an HTTP proxy for the message *(page 76; either http (the default) or https is configured for the Network Access Point Type)*.

- **Regarding claim 31**, Sun discloses the computing apparatus of claim 20, wherein the client application comprises a JAVA compatible client application *(pages 34-39 and 91-118; Figs. 1-2 and 1-3; Java client and JAX-RPC are Java based applications)*.

- **Regarding claim 32**, Sun discloses the computing apparatus of claim 20, wherein the Web service client proxy comprises:

a deployable Web service client proxy *(pages 95-99; Generating a JAX-RPC Client; JAX-RPC Client is known as a deployable Web service client)*.

- **Regarding claim 33**, Sun discloses the computing apparatus of claim 20, wherein the Web service client proxy comprises:

a standalone Web service client proxy *(pages 110-111; generating a kSOAP Client; kSOAP Client is known as standalone Web service client)*.

- **Regarding claim 34**, Sun discloses a system comprising:

a first node having a Web service to exchange a message with a Web service client *(pages 31-39; Figs. 1-1 to 1-3; Web container, known as a first node, exchanges SOAP messages with the client where SOAP client proxy is running)*; and

a second node coupled with the first node, the second node having the Web service client *(the client, where SOAP client proxy is running, functions as a second node)* including:

a client application to invoke a method of the Web service *(pages 91-122; Figs. 3-1 to 3-9; web client, created from either a local IDE web service, from WSDL, or from a UDDI registry entry, sends SOAP request to the web server),*

a Web service client proxy coupled with the client application to expose the method of the Web service to the client application and exchange the message with the Web service *(pages 91-122; Figs. 3-1 to 3-9; JAX-RPC Client is created and deployed; SOAP messages are exchanged between Web server and Web client)*, and

a protocol implementation coupled with the Web service proxy *(page 21; Sun One Studio 5 IDE to build web services, to make web services available to others through a UDDI registry, and to generate web service client from a local web service or a UDDI registry; page 58; the IDE provides comprehensive support for creating web service clients from any of the following sources: a local IDE web service, a WSDL file, a web service published in a UDDI registry)* to process the message exchanged between the Web service client proxy and the Web service *(pages 34-38 and 91-122; Figs. 1-1, 1-2, and 1-3; Web client sends SOAP request message to the Web server and receives SOAP response from the Web server),* the protocol implementation comprising a security protocol implementation to provide a security service for the message *(pages 189-214; Fig. A-2; either HTTP Basic Authentication or HTTPS/SSL Authentication or Encryption is utilized to secure message exchanged between client and server).*

- **Regarding claim 36**, Sun discloses the system of claim 34, wherein the protocol implementation further comprises a wrapper protocol implementation to provide a wrapper

for the message *(page 36; Fig. 1-2; SOAP request is an XML wrapper that contains a method call and SOAP response is an XML wrapper that contains the return value)*.

- **Regarding claim 37**, Sun discloses the system of claim 34, wherein the protocol implementation further comprises a session protocol implementation to process a session between the Web service client and the Web service *(pages 61-63; Figs. 2-15, 2-16, 2-17, and 2-18; session bean stateful can be set by using either Session EJB setup dialog box; Figs. 2-15 and 2-16, or modifying the session bean class to add methods named startOrder and submitOrder, Figs. 2-17 and 2-18)*.

- **Regarding claim 38**, Sun discloses the system of claim 34, wherein the protocol implementation further comprises an HyperText Transport Protocol (HTTP) protocol implementation to establish an HTTP proxy for the message *(pages 189-214; Fig. A-2, Authentication setup dialog box; JAX-RPC and IDE support HTTP Basic Authentication and HTTPS/SSL Authentication and Encryption; check the radio button for selecting HTTP Basic Authentication)*.

- **Regarding claim 40**, Sun discloses the system of claim 34, wherein at least one of the first node and the second node is an application server *(pages 31-39; Figs. 1-1 to 1-3; Web container functions as Web server, where JAX-RPC is running)*.

- **Regarding claim 41**, Sun discloses an application server comprising:

a client application to invoke a method of a Web service *(pages 91-122; Figs. 3-1 to 3-9; web client, created from either a local IDE web service, from WSDL, or from a UDDI registry entry, sends SOAP request to the web server)*;

a Web service client proxy coupled with the client application to expose the method of the Web service to the client application and exchange the message with the Web service *(pages 91-122; Figs. 3-1 to 3-9; JAX-RPC Client is created and deployed; SOAP messages are exchanged between Web server and Web client)*;

a protocol implementation coupled with the Web service client proxy *(page 21; Sun One Studio 5 IDE to build web services, to make web services available to others through a UDDI registry, and to generate web service client from a local web service or a UDDI registry; page 58; the IDE provides comprehensive support for creating web service clients from any of the following sources: a local IDE web service, a WSDL file, a web service published in a UDDI registry)* to process the message exchanged between the Web service client proxy and the Web service *(pages 34-38 and 91-122; Figs. 1-1, 1-2, and 1-3; Web client sends SOAP request message to the Web server and receives SOAP response from the Web server)*; and

a means for setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy *(page 95-105; Figs. 3-7 and 3-8; user is able to set the client type, either JAX-RPC or kSOAP; user is able to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components)*.

- **Regarding claim 42**, Sun discloses the application server of claim 41, wherein the means for setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy comprises:

a means for selecting an authentication type for a client authentication protocol implementation to define an authentication type for the message exchanged between the Web service client proxy and the Web service *(pages 189-214; Fig. A-2, Authentication setup dialog box; JAX-RPC and IDE support HTTP Basic Authentication and HTTPS/SSL Authentication and Encryption; user can select the radio button to set HTTP Basic Authentication)*.

- **Regarding claim 43**, Sun discloses the application server of claim 41, wherein the means for setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy comprises:

a means for selecting a HyperText Transport Protocol (HTTP) proxy for a client HTTP proxy protocol implementation to define an HTTP proxy for the message exchanged between the Web service client proxy and the Web service *(pages 76-77; Fig. 2-29; user is able to select http (the default) or https for the Network Access Point Type using UDDI publish Web service dialog box)*.

- **Regarding claim 44**, Sun discloses the application server of claim 41, wherein the means for setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy comprises:

a means for selecting a wrapper for a client wrapper protocol implementation to define a wrapper for the message exchanged between the Web service client proxy and the Web service *(page 36, and 153-187; Fig. 1-2; SOAP request is an XML wrapper that contains a method call and SOAP response is an XML wrapper that contains the return value; XML operation is used to encapsulate a number of business methods)*.

- **Regarding claim 45**, Sun discloses an article of manufacture comprising:

an electronically accessible medium providing instructions that, when executed by an apparatus, cause the apparatus to access a description of a Web service *(pages 91-129; Web service client is created based on the WSDL file of a Web service)*;

generate a Web service client proxy based, at least in part, on the accessed description of the Web service *(pages 112-122; sections Creating a Client From WSDL and Creating a Client From a UDDI Registry Entry; the default URL comes from the WSDL that is used to generate the client proxy)*;

provide a client protocol implementation for the generated Web service client proxy, wherein the provided client protocol implementation is to process a message exchanged between the Web service client proxy and the Web service *(pages 29 and 112-122; WDSL file describes external interface for a Web service; with this information, developers can create SOAP client capable of issuing remote requests to the Web service; tModels is published by web service provider to provide external interface to create web service client)*; and

configure a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy *(page 95-105; Figs. 3-7 and 3-8; user is able to set the client type, either JAX-RPC or kSOAP; user is able to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components)*.

- **Regarding claim 46**, Sun discloses the article of manufacture of claim 45, wherein the instructions that, when executed by the apparatus, cause the apparatus to configure a feature of the client protocol implementation include instructions that cause the apparatus to configure an authentication type for a client authentication protocol implementation to define an authentication type for the message *(pages 189-213; Figs. A-2, A-3; Authentication Dialog box is used to setup Authentication)*.

- **Regarding claim 47**, Sun discloses the article of manufacture of claim 45, wherein the instructions that, when executed by the apparatus, cause the apparatus to configure a feature of the client protocol implementation include instructions that cause the apparatus to configure a HyperText Transport Protocol (HTTP) proxy for a client HTTP proxy protocol implementation to define an HTTP proxy for the message *(pages 76-77; Fig. 2-29; either http (the default) or https for the is selected and implemented for communication between Web server and Web client)*.

- **Regarding claim 48**, Sun discloses the article of manufacture of claim 45, wherein the instructions that, when executed by the apparatus, cause the apparatus to configure a feature of the client protocol implementation further comprise instructions that cause the apparatus to configure a wrapper type for a client wrapper protocol implementation to define a wrapper type for the message *(page 36; Fig. 1-2; SOAP request is an XML wrapper that contains a method call and SOAP response is an XML wrapper that contains the return value)*.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

7. **Claim 30 and 39 are rejected under 35 U.S.C. 103(a)** as being unpatentable over Sun Microsystems (hereinafter "Sun"), "Building Web Services – Sun™ ONE Studio 5 programming Series," published in June 2003 and further in view of Sun Microsystems (hereinafter "Sun-ONE"), "Sun One Architecture Guide – Delivering Services on Demand," published in 2002.

- **Regarding claim 30**, Sun discloses the computing apparatus of claim 20.

Sun does not explicitly disclose the protocol implementation is a pluggable protocol implementation.

However, Sun-ONE discloses a Sun ONE architecture, wherein the protocol implementation is a pluggable protocol implementation (*pages 185-186 and 211; pluggable protocol adapters, which are connection component, are available with in the Sun ONE architecture to provide model and protocol mapping between applications in the presentation tier and services in the management services tier; The Java Authentication and Authorization Service (JAAS) implements a Java version of the Standard Pluggable Authentication Module (PAM) frame work*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the method of Sun with that of Sun-ONE to include the protocol implementation is a pluggable protocol implementation to provide the model and protocol mapping between application in the presentation tier and services in the management services tier (*pages 185-186*).

- **Regarding claim 39**, claim 39 is similar in scope to claim 30, and is therefore rejected under similar rationale.

Response to Applicant's arguments

8. Applicants' arguments, filed on 01/28/2008, have been fully considered but they are not persuasive.

9. **Applicants' arguments:**

a. <u>Rejections of claims 1-19 and 41-48 under 35 U.S.C. 102(a)</u>:

- "the Office Action therefore, appears to equate changing the Generate Presentation property of the client node with setting a feature of the **client protocol**

implementation as claimed"; "the '***client node***' described by Sun Web Services is not the same as the '***client protocol implementation***'."

- "The reference does not disclose 'setting a feature of the **client protocol implementation** … without regenerating the **Web service client proxy**'."

b. Rejections of claims 20-29, 31-38, and 40 under 35 U.S.C 102 (a):

- "the cited reference fails to disclose 'a protocol implementation **coupled with the Web service client proxy … comprising a security protocol implementation**'."
- "the reference does not teach or suggest that the security technologies are comprised within a protocol implementation."

c. Rejections of claims 30 and 39 under 35 U.S.C 103 (a).

- "Sun Web Services and Sun SDOD fail to disclose a protocol implementation coupled with the Web service client proxy … comprising a security protocol implementation."

10. **The Examiner disagrees for the following reasons**:

a. Per (a):

- As described in chapter 3 "Creating a Web Service Client," *(page 91-129)*, after having successfully created a client from a local Web service, user is able to ***set the client type***, either JAX-RPC or kSOAP by selecting appropriate value for SOAP runtime property *(page 93-94)*. In addition, user is able to set the generate presentation property value to 'false' to keep custom change and generates a stateless client *(page 95-105; "if the value is changed to false,* ***only the client proxy classes***

are created *when generating client components" and "if you manually change the client's conversational property to 'false' and then use the Generate Client Files,* ***the IDE keeps your change and generates a stateless client.****").* (emphasis added);

- As explained above, Sun Web Services discloses "setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy." *(page 95-105; user is able to* ***set the client type****, either JAX-RPC or kSOAP; user is able* ***to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components****).* (emphasis added).

b. Per (b):

- Sun Web Services discloses "a protocol implementation coupled with the Web service client proxy … comprising a security protocol implementation," as following: a protocol implementation coupled with the Web service proxy *(page 21; Sun One Studio 5 IDE to build web services, to make web services available to others through a UDDI registry, and to generate web service client from a local web service or a UDDI registry; page 58; the IDE provides comprehensive support for creating web service clients from any of the following sources: a local IDE web service, a WSDL file, a web service published in a UDDI registry)* to process a message exchanged between the Web service client proxy and the Web service *(pages 34-38 and 91-122; Figs. 1-1, 1-2, and 1-3; Web client sends SOAP request message to the Web server and receives SOAP response from the Web server),* the protocol implementation comprising a security protocol implementation *(pages 189-214; Fig. A-2; either*

HTTP Basic Authentication or HTTPS/SSL Authentication or Encryption is utilized to secure message exchanged between client and server).

- Sun Web Service suggests the use of security for the implementation as following: the security technologies are comprised within a protocol implementation *(pages 189-214; Fig. A-2; either HTTP Basic Authentication or HTTPS/SSL Authentication or Encryption is utilized to secure message exchanged between client and server).*

c. **Per (c):**

- As discussed in sections (a) and (b) above, Sun Web Services discloses "setting a feature of the client protocol implementation to define a behavior of the Web service without regenerating the Web service client proxy." *(page 95-105; user is able to **set the client type**, either JAX-RPC or kSOAP; user is able **to set the Generate Presentation property value to 'false' to keep custom changes and generates a stateless client; only the client proxy classes are created when generating client components**).* (emphasis added).

Conclusion

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Luu Pham whose telephone number is 571-270-5002. The examiner can normally be reached on Monday through Friday, 7:30 AM - 5:00 PM (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Emmanuel L. Moise can be reached on 571-272-3865. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Luu Pham/
Examiner, Art Unit 2137

/Emmanuel L. Moise/
Supervisory Patent Examiner, Art Unit 2137